# Efficient, Dense, Object-based Segmentation from RGBD Video

Mahsa Ghafarianzadeh[1], Matthew B. Blaschko[2] and Gabe Sibley[1]

*Abstract*— **Spatio-temporal cues offer a rich source of information for inferring structural and semantic scene properties. A particularly useful representation in computer vision is a spatio-temporal video segmentation. Together with motion, knowledge of depth can substantially improve superpixel segmentation. In this work we present a novel framework for spatio-temporal segmentation from RGBD video. The method employs both low-level (intensity, color) and high-level (deformable parts model) appearance features. Motion is incorporated through the use of optical flow to construct the temporal connections in the graph Laplacian. Depth cues are incorporated in the similarity metric to provide an informative cue for object boundaries at depth disparities. Naïve application of spectral clustering to dense spatio-temporal graphs leads to a high computational cost that is typically addressed through the use of GPUs or computer clusters. By contrast, we build upon a recently proposed Nyström approximation strategy for spatio-temporal clustering that enables computation on a single core. We further explore structured local connectivity patterns to give high performance at low computational cost. Also we propose a novel context-aware aggregation method that uses a deformable parts model to group the detected parts of the object as a single segment with an accurate boundary. Detailed experiments on the NYU Depth Dataset and TUM RGBD Dataset is performed to compare against previous large-scale graph-based spatio-temporal segmentation techniques which shows the substantial performance advantages of our framework.**

## I. Introduction

Video segmentation is defined as the problem of grouping a sequence of images into coherent regions with similar appearance, motion and temporal continuity. The goal of video segmentation is to represent image sequences through homogeneous regions where the same object should carry the same unique label along the whole video. Spatio-temporal cues are essential to obtain good performance in video segmentation, as appearance-based cues may be ambiguous in regions of low texture and similarity between foreground and background. An important additional information source has come in the form of RGBD cameras. These cameras have become ubiquitous in recent years, and are being incorporated in a large variety of entertainment systems, user interfaces, robotic sensors, etc. The information provided by depth is *complementary* to that provided by motion cues: motion can provide information about points that (do not) move together, while depth sensors provide information about depth disparities even in the absence of motion. This provides an excellent opportunity to boost spatio-temporal segmentation accuracy through the use of potentially noisy depth information. Spatio-temporal segmentation of RGBD videos has numerous applications in sub-fields of computer vision. In robotics, RGBD cameras are more-and-more frequently employed to infer environment structure and to identify individual objects that can be manipulated [2]. Self-driving cars must be able to accurately determine scene components, their spatial extents, and their trajectories [21]. Consequently analysis of RGBD images and videos is becoming an important problem in robotics [11].

High level vision tasks often require, or can be aided by knowledge of the spatio-temporal relationship between objects inherent in a high-quality RGBD video segmentation. In object tracking, frequent difficulties such as occlusion handling can be alleviated through segmentations that account for both appearance motion cues and depth disparities. Action recognition has been shown to be substantially aided through the incorporation of depth cues [18] and spatio-temporal segmentations with depth information is a rich representation for this task.

In this paper, we show that the inclusion of depth and deformable part-model information can greatly improve video segmentation. We are interested in unsupervised spatio-temporal segmentation of the RGBD video sequence. Our approach extends prior work on graph-based spatio-temporal segmenation [13] by adding depth information, parts-based aggregation, and also a novel technique for connecting pixels within a neighborhood. The central strategy is to build a pixel level spatio-temporal graph that encodes appearance, depth and motion information with temporal links based on optical flow and then cut the graph to obtain segments that are coherent over time. We present two different aggregation methods for merging the over-segmentation result. One is a hierarchical approach and the other is based on the deformable part models for an object of interest. We employ a recently proposed Nyström approximation strategy to ensure computational tractability, resulting in a method that is both accurate, and tractable to run on a single core. We believe this to be the first demonstration of dense spatio-temporal segmentation from RGBD videos that does not make use of extensive computational infrastructure such as GPUs or compute clusters. At the same time, we demonstrate substantially improved performance over a variety of approaches. Advantages of the proposed method are demonstrated using the NYU Depth Dataset [22] and the TUM RGBD Dataset [27] as well as on our own dataset. This paper makes three main contributions: (i) a

*This work is partially funded by Internal Funds KU Leuven and the European Commission through FP7-MC-CIG 334380.

[1] Mahsa Ghafarianzadeh and Gabe Sibley are with the Department of Computer Science, University of Colorado-Boulder, Boulder, CO 80309, USA mahsa.ghafarianzadeh|gsibley@colorado.edu

[2] Matthew B. Blaschko is with the Center for Processing Speech and Images, KU Leuven, Kasteelpark Arenberg, 3001 Leuven, Belgium matthew.blaschko@esat.kuleuven.be

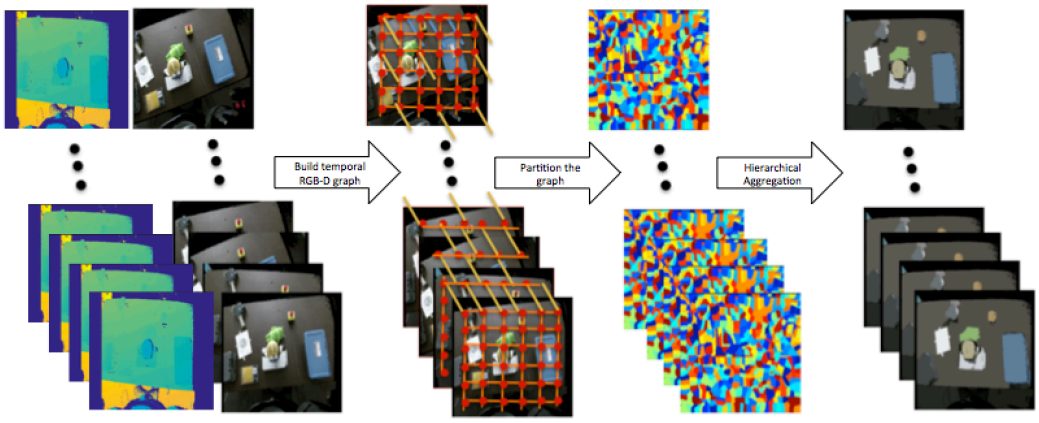


Fig. 1. Approach Overview: (left-to-right) Given a sequence of RGBD images, We first construct a similarity graph using appearance and depth information, we also add the temporal links computed using optical flow, then we use Nyström method to obtain an over-segmentation of the images, and finally we use a hierarchical aggregation method to obtain the final spatio-temporal segments in the video

novel extension of a spatio-temporal graph Laplacian based spectral clustering technique is introduced that incorporates depth information, (ii) a randomized strategy is proposed for the structure of a pixel neighborhood in the construction of the spatio-temporal graph Laplacian, and (iii) a context aware aggregation strategy that employs an object detection system is employed to find object-centric segmentations.

## II. Related Work

Techniques for spatio-temporal segmentation can generally be grouped into several categories based on the assumptions made by the approach. One category of techniques is based on tracking and clustering a sparse number of point trajectories (up to 4% of the pixels) into temporally coherent moving regions [23], [4]. Another category is dense video segmentation in which the video is presented as a spatio-temporal volume. One popular method is hierarchical graph-based segmentation [14], which builds on graph-based image segmentation proposed by [10], in which a volumetric video graph is over-segmented into spatio-temporal regions. These regions form a region graph which is used to merge similar regions repeatedly into larger regions and creates a hierarchical segmentation of the video. In [7], hierarchical mean shift analysis is introduced, which maps pixels of a video stack into a 7D feature space and then uses mean shift repeatedly to achieve hierarchical clustering of the points in the video. Another recent work [5], uses a generative probabilistic model to find temporally consistent superpixels in a video stream. Normalized cuts for graph partitioning has been widely used in both image and video segmentation [25]. Since the underlying spectral clustering is computationally expensive, it is often impractical to use for large scale data. The Nyström method [31], [12], is a numerical solution for approximating p.s.d. matrices, which can be used to speed up spectral clustering. In [13], an efficient method is applied to randomly select a subset of pixels and estimate a rank-$k$ approximation of the similarity matrix which enables the performance of spatio-temporal segmentation in memory on a single CPU. In [28], GPU clusters are necessary to perform spectral clustering naïvely on the original, unapproximated Laplacian. These described methods are based on only appearance cues.

In [30] a depth-adaptive supervoxel video segmentation technique is proposed that uses color and depth information to compute the supervoxels and then applies spectral clustering to partition them into spatio-temporal segments. Another algorithm [1] uses a GPU based parallel Metropolis algorithm to combine color and depth information and segment the images. They use real-time optical flow to warp obtain segment labels and track them to the next frame. Also in [16], an efficient hierarchical graph based segmentation is presented for segmenting 3D RGBD point clouds. They consider a moving window over a point cloud that segments the similar points into regions which later are merged together repeatedly. A bipartite graph matching is used to obtain the final segmentation at a given level of the hierarchy. Other methods for 3D segmentation of point clouds such as [17], [24], [26] do not consider time and are focused on static scenes, or require strong supervision [15].

## III. Spatio-Temporal Segmentation

Our approach has three main steps, as shown in Figure 1. First, we randomly sample a subset of pixels in the video sequence and form a sparse affinity matrix, which is later used to estimate a low rank approximation of the full affinity matrix. Next, we perform spectral clustering on the affinity matrix to obtain an over-segmentation of the sequence. Finally, we apply ultra-metric contour maps [3] on the resulting over-segmentation to obtain a smaller number of regions.

### A. Spatio-temporal Graph

For each frame in an RGBD video sequence, we create a similarity graph $G = (V, E, W)$, where each pixel is a vertex $v_i \in V$ and is connected to $v_j$ by an edge $e_{ij}$ if they are within distance $r$ from each other. The edge $e_{ij}$ is weighted by $w_{ij}$ that shows the similarity between pixels $v_i$ and $v_j$.

$W$ is defined as the following:

$$w_{ij} = \exp\left(\frac{-d^2(p_i, p_j)}{\sigma_p^2}\right) \quad (1)$$

Where $w_{ij} = 0$ for $i = j$, and $d(p_i, p_j)$ is a linear combination of the difference in intensity, color and spatial position of two pixels:

$$d(p_i, p_j) = \alpha \times d_c(p_i, p_j) + \beta \times d_l(p_i, p_j) \quad (2)$$

Also $\sigma_p$ is a scaling factor, and we set $\sigma_p = 0.01$, $\alpha = 0.4$ and $\beta = 0.6$ *a priori*. $d_c$ is the color distance of pixels $p_i$ and $p_j$, measured by the euclidean distance in the CIELAB color space, and $d_l$ is the euclidean distance between 3D position of $p_i$ and $p_j$ in camera coordinates, which is computed using a pinhole camera model:

$$xyz_p = depth_p \times (\frac{x - c_x}{f}, \frac{y - c_y}{f}, 1)^T \quad (3)$$

where $x$ and $y$ are the position of pixel $p$ in image plane and $f$, $c_x$ and $c_y$ are the camera parameters. As mentioned earlier, we only measure the similarity of the pixels within distance $r$ in the image plane due to the fact that it is more likely that nearby points in 3D with similar colors belong to the same region. Choosing the graph radius $r$, is a tradeoff between segmentation quality and computation cost. As discussed in [6], larger $r$ makes the quality of segmentation better and smaller $r$ makes the segmentation faster. If we consider a neighborhood around pixel $p$, the graph weights have low variance, if there is no edge falling between any two adjacent pixels in the neighborhood it therefore is redundant information. By considering larger $r$, the chances of having an edge in the neighborhood increases which leads to detecting contours and propagating the local cues across larger images regions, however the graph becomes denser and more expensive to compute. We alleviate this trade-off by considering a mid-range sparse neighborhood around pixels to benefit from grouping information of a larger area while low cost computation of the sparse neighborhood (Figure 2 and Figure 4, implementation details are provided in Section IV). The random pixel-neighborhood pattern in the third column of Figure 4 gives an empirically superior balance between computational cost and accuracy than either a structured pattern or a dense neighborhood connectivity.

Prior to computing the similarity between the pixels, we use a joint bilateral filter [29], [8] for smoothing the image and removing the noise in color values. The joint bilateral filter controls the smoothing weight for color image with regards to the depth image and prevents color from flowing over the depth boundaries. The joint bilateral filter is defined as:

$$I_{p_c} = \frac{1}{k}\Sigma g_c(p_c - p\prime_c) g_d(p_d - p\prime_d) I\prime_{p_c} \quad (4)$$

where

$$k = \Sigma g_c(p_c - p\prime_c) g_d(p_d - p\prime_d) \quad (5)$$

is a normalization term and $p_c$ is the color value, $p_d$ is the depth value and $g$ denotes the Gaussian kernel. Here, kernel $g_c$ sets the weights based on the color difference of the pixels,

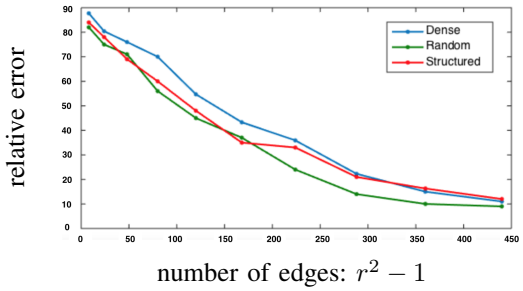


Fig. 2. The error based on choice of neighborhood size $r$ = 3, 5, 7, 9, 11, 13, 15, 17, 19 and 21 for structured pattern, random pattern and dense pattern is shown in red, green and blue respectively. Note that the error is reported for the number of edges between the center pixel and the pixels in the neighborhood $r$. The random pixel neighborhood strategy consistently gives the best performance (cf. Figure 4).

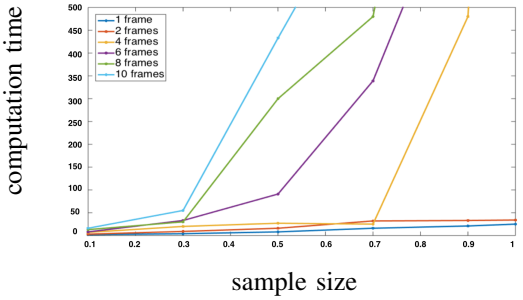


Fig. 3. Computation time for eigen-decomposition of the normalized Laplacian is shown for different sample sizes in efficient Nyström method. Different colors represent different number of frames used to build the graph. (cf. Figure 5).

while the edge stopping kernel $g_d$ computes proper weights for nearby pixels based on the depth differences.

After creating the graph for each frame in the RGBD sequence, we need to add the motion information and extend the graph to the spatio-temporal case. In this step, we use optical flow [20] to compute the motion vectors between frames. Since the optical flow vector implies that the source pixel at time $t-1$ belongs to the same region as the destination pixel in time $t$, we connect pixel $(x, y)$ in frame $t$ to its 9 neighbors along the backward flow $(u, v)$ in frame $t-1$, *e.g.* $(x + u(x, y) + \delta_x, y + v(x, y) + \delta_y)$ for $\delta_x, \delta_y \in \{-1, 0, 1\}$, and then we compute the similarities between the connected pixels using Equation (1).

### B. Over-segmentation

Given the similarity graph $G$, we form the affinity matrix $W$ which is a sparse symmetric band matrix of size $n \times n$, $n$ is the total number of pixels in the video. $W$ has the following structure:

$$W = \begin{vmatrix} W_{11} & W_{12} & 0 & 0 & \dots & 0 \\ W_{21} & W_{22} & W_{23} & 0 & \dots & 0 \\ 0 & W_{32} & W_{33} & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & W_{nn} \end{vmatrix} \quad (6)$$

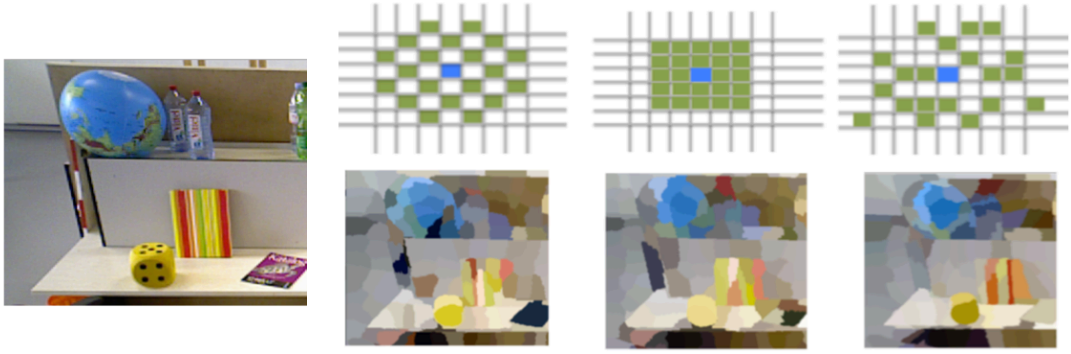

Fig. 4. Random, dense, and structured pixel neighborhood patterns and the resulting spatio-temporal segmentations. In all cases, the number of edges, and thus the computational complexity, are the same.

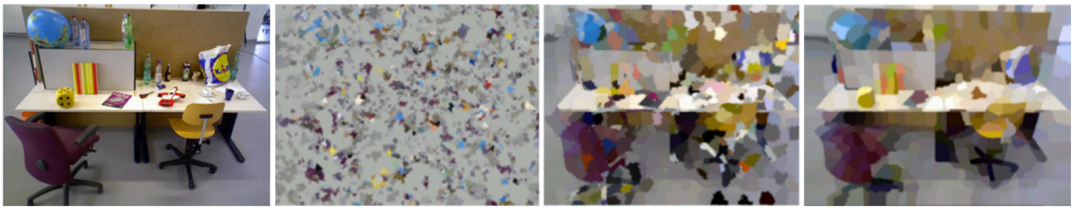

Fig. 5. Over-segmentation quality for different sample size in Nyström method: original frame, sample size: 10%, 30% and 50%

where $W_{i,i}$ denotes the affinity matrix between pixels in frame $i$ and $W_{i,j}$ denotes the affinity matrix between the pixels in frame $i$ and frame $j$. After building the spatio-temporal graph for all of the pixels in the video, which encodes appearance,depth and motion information, the spectral clustering algorithm is applied to partition the graph and obtain the over-segmentation of the video. In short, for spectral clustering, first we need to compute the normalized graph Laplacian of $W$ by $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ where $D$ is the diagonal degree matrix defined as $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Then an eigen-solver is used to find the eigenvectors corresponding to the $k$ largest eigenvalues of the normalized Laplacian matrix. The complexity of the eigen decomposition of $L$ is $O(n^3)$ which makes it very expensive and impractical for large datasets like videos. Therefore we apply a time and space efficient Nyström method [19] similar to [13] that has a complexity of $O(nmk)$. The idea is to randomly select a subset of $m$ pixels in the video and form the affinity matrix for these selected pixels. Then the normalized Laplacian matrix $M$ is computed and used to estimate the eigenvectors of a rank-$k$ approximation of the normalized Laplacian matrix $L$ (Figure 5) for the entire set of pixels in the video. Finally another clustering algorithm such as $k$-means is used to cluster the estimated eigenvectors of the normalized Laplacian $L$. This will give us the cluster assignment for all of the pixels in the video. Further details are provided in [13]. This gives an over-segmentation of the video in the form of 3D superpixels that are coherent in terms of appearance and position in space and time. Note that in the context of this paper, the term superpixel has been used interchangeably with the term 3D superpixel.

### C. Hierarchical Aggregation

The over-segmentation of the RGBD video produces 3D depth-labeled superpixels. For each of these superpixels we compute a feature vector $f$. The components of $f$ are the superpixel LAB color histogram $s_c$, superpixel mean surface normal $s_n$ and superpixel centroid $s_{xy}$. We start by defining a graph structure where each superpixel is a vertex in the graph and is connected to all neighboring superpixels with edges. The edges are weighted by the similarity between connected superpixels, based on the fact that similar superpixels have similar appearance, are spatially close and have normals that point to the same direction. We measure the similarity of neighboring superpixels $s_i$ and $s_j$ by:

$$S_{ij} = \exp\left(\frac{-d^2(s_i, s_j)}{\sigma_s^2}\right) \tag{7}$$

where

$$d(s_i, s_j) = \gamma \times d_c(s_i, s_j) + \delta \times d_n(s_i, sj) + \eta \times d_{xy}(s_i, s_j) \tag{8}$$

$d_c(s_i, s_j)$ is the $\chi^2$ distance between the color histograms, $d_n(s_i, s_j)$ measures the angle between surface normals, $d_{xy}(s_i, s_j)$ is the Euclidean distance of the superpixels centroids and $\sigma_s$ is a scaling factor. Also we set $\gamma = 0.4$, $\delta = 0.4$, $\eta = 0.2$ and $\sigma_s = 0.01$ *a priori*. Here, the intuition is that we want the superpixels that have similar color and surface normal to be grouped together but we put less emphasis on where in the image they are located (for example superpixels of a background wall that is occluded with a foreground object should be grouped into one segment). After creating the neighborhood graph for superpixels, we use an approach similar to the ultra-metric

Fig. 6. Hierarchical Aggregation using an Ultra-Metric Contour Map approach is shown. Left column is the original frame taken from TUM RGBD Dataset [27], the second to fourth columns show different levels of segmentation in the hierarchy with maximum number of clusters: 400, 100, and 50 respectively

contour map [3] to merge the superpixels repeatedly. We sort the edges in the neighborhood graph and start from the edge with highest value of similarity, and merge the two corresponding superpixels. Then we update the weights and repeat this process until there are no more edges. Also it is possible to make this process faster by using a bucket sort list in which we go thorough all the edges in one bucket and merge the corresponding superpixels before updating the neighborhood graph. This process gives us a hierarchy of segmentation in a bottom-up approach. We can obtain the final segmentation by using a threshold to stop merging the superpixels either when it reaches a certain level in the hierarchy or when a desired number of segments is obtained. Figure 6 presents the hierarchical aggregation results for different levels of segmentation.

### D. Parts-Based Aggregation

In this section, we introduce a novel method for merging the superpixels obtained from the over-segmentation by applying an object detection technique [9] based on mixtures of multiscale deformable part models. These models are trained using a discriminative process that requires bounding boxes. A DPM model for an object with $n$ parts is defined by a root filter and $n$ part filters. To detect an object in the image, an overall score for each root location according to the best placement of the parts is computed. High-scoring root locations define detections while the locations of the parts that yield a high-scoring root location define a full object hypothesis with a bounding box. Given these bounding boxes and the filter scores, we want to segment out the object from the background. To do this, for each superpixel inside the bounding box, we first find the mean value of the scores given by the filter responses $s_{dpm}$. Then we build the superpixel neighborhood graph and compute the edge weights using Equation (7) where

$$d(s_i, s_j) = \mu \times d_{dpm}(s_i, s_j) + \nu \times max_{x \in B}(edge(x)) \quad (9)$$

$d_{dpm}(s_i, s_j)$ is simply the difference between superpixel scores and $edge(x)$ is the edge strength at pixel $x$ on the boundary of two superpixels. We set $\mu = 0.6$ and $\nu = 0.4$ *a priori*. If the filter responses for two adjacent superpixels are high enough, we still want to merge them even if there is an edge between them, so we set a higher value for $\mu$. We also applied a Sobel operator to detect the edges using the gradient of the image. Then the same approach as the previous section is used to merge the superpixels repeatedly. This method of aggregation can be used if one is interested in tracking a boundary accurate object in contrast to bounding box methods. To evaluate our results in this paper, however, we only used the hierarchical aggregation to obtain a dense segmentation of the sequence. Figure 7 shows the difference between hierarchical aggregation and parts-based aggregation.

## IV. Results

Several comparative experiments were run to assess the performance of the proposed method using publicly available RGB-D video sequences. In Table I, 'RGB-D Video' indicates the proposed method. The second method called 'Depth two-stage' uses the two-stage strategy proposed in [16] to incorporate depth information in the spatio-temporal segmentation, while still using the same appearance based cues as our proposed method. The fundamental difference between this method and the proposed method is that instead of combining depth and color information linearly, first a segmentation of a sequence of frames using depth and motion information is obtained, and then an over-segmentation of the frames is done using color and motion information while respecting the depth boundaries from previous step. The third method called 'RGB Video [13]' uses similar appearance cues to those employed in our method, but does not take depth information into account. The fourth method called 'RGB Video [14]' uses a similar approach to the graph based image segmentation 'RGB Image [10]', but extends it to the video setting by adding temporal information.

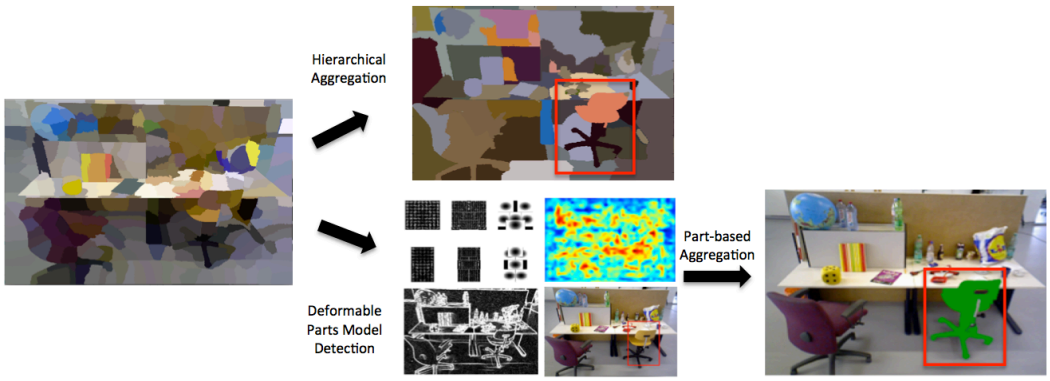


Fig. 7. Hierarchical Aggregation vs. Parts-Based Aggregation Approach: (Top-row from left to right) shows the segmentation result for the chair using hierarchical aggregation, (Bottom-row from left to right) the root and part filters for "chair", the filter score for chair detection, detected edges using Sobel operator, prediction of boundary box and result of aggregating superpixels in the bounding box. Without the use of the context aware aggregation, the different articulated parts of the chair are misidentified as belonging to different objects. The incorporation of the parts based model enables the segmentation framework to correctly group these parts as a single object.

These results show that combining depth information with color and temporal cues improves the segmentation result significantly and having an initial depth segmentation is not necessary. The presented method tends to maintain the depth boundaries of the objects and avoids merging them into other parts of the scene. Three further experiments have been performed: two sequences of size 100 frames and one sequence of size 200 frames have been chosen from NYU Depth Dataset [22] and one sequence of size 150 frames has been chosen from our own dataset Fig. 8. The qualitative results shown in Figures 4, 5 and 6 are sequences chosen from TUM RGBD Dataset [27]. Table I shows the detailed numerical comparison between our method and the rest of methods. Here, the density of all the methods is 100% which denotes the percentage of pixels for which a label is reported. The overall error is the number of mislabeled pixels over the total number of labeled pixels. The overall accuracy is the number of correctly labeled pixels over the total number of pixels. All pixels covering their assigned region in ground truth are counted as correctly labeled pixels, all others count as mislabeled pixels. The average error is similar to the overall error but computed separately for each region first and then the average across all the regions is reported. The over-segmentation error indicates the number of clusters that need to be merged to fit the ground truth regions. Also the number of regions with less than 10% error is reported as the extracted objects. The same evaluation method has been previously used in [13], [4]. We can see that an integrated system incorporating RGB-D video gives the best results by a substantial margin, with an approximately 1/3 reduction in error rate vs. the best performing video segmentation competitor [13]. Furthermore, we show that the direct inclusion of depth in the graph segmentation reduces the error rate by a further 16% or more as compared with our implementation that uses the two-stage method proposed in [16]. Also Figure 2 and 4, compare the error for different graph radius sizes and patterns. Observe that the best result is achieved by choosing a larger radius with a sparse random pattern. Note, in all of our experiments, the graph radius is set to $r = 5$, which means that the center pixel is connected to 120 pixels in an 11x11 neighborhood (e.g. in the dense pattern). For both structured and random neighborhood pattern, the center pixel is connected to the same number of pixels as dense pattern but each has a different structure as shown in Fig. 4. In all of the experiments in this paper we use the random neighborhood pattern to build the spatio-temporal graph. Since a random subset of pixels is used in the video to approximate the normalized Laplacian matrix $L$, in Figure 3 and 5 the overall computation time and accuracy is reported for different sample sizes. It is obvious that selecting more pixels improves the segmentation result, however there is a trade-off between accuracy and computation cost. For all the experiments in this paper a random subset of 50% of pixels is chosen. On average, the runtime for the segmentation was about 20 minutes for 100 frames of size $640 \times 480$ on a single CPU. Finally, Figure 8 presents a qualitative comparison between the proposed method and other state-of-the-art methods based on superpixel segmentation.

## V. Discussion

Experiments systematically show an improvement using the proposed framework, both from a quantitative, and from a qualitative perspective. Results show a reduction in the overall error rate by approximately one third on both the first and second experiments. Of high importance to unsupervised object detection, the number of extracted objects is increased in both experiments as well. On the second experiment, the average error increases by one percent, but the first experiment shows an average error decrease of approximately one quarter. This disparity can be accounted for by the different distribution of segment sizes in the two sequences, and the overall error is consistently decreased by our method while using only 50% of randomly chosen pixels from the entire video. The random pixel neighborhood pattern gave the best result, and did so at a reduced computational cost

TABLE I

COMPARISON BETWEEN THE PROPOSED METHOD AND EXISTING METHODS IN THE LITERATURE.

| Method | Overall Error | Overall Accuracy | Average Error | Over-segmentation | Extracted Objects |
|---|---|---|---|---|---|
| 2 sequences(100 frames) | | | | | |
| RGB-D Video | **9.8**% | **90.2**% | **15.1**% | 14 | 16 |
| Depth two-stage | 11.7% | 88.3% | 16.7% | 15.5 | 16 |
| RGB Video [13] | 14.3% | 85.7% | 17.2% | 19.2 | 14.3 |
| RGB Video [14] | 16.4% | 83.6% | 19.1% | 18.3 | 13 |
| RGB Image [10] | 21.7% | 78.3% | 23.7% | 24 | 10 |
| one sequence(150 frames) | | | | | |
| RGB-D Video | **11.4**% | **88.6**% | **17.4**% | 16.8 | 12.4 |
| Depth two-stage | 13.9% | 86.1% | 20.1% | 18 | 10.7 |
| RGB Video [13] | 16.3% | 83.7% | 21.4% | 19.2 | 9.3 |
| RGB Video [14] | 16.1% | 84.9% | 21.8% | 18.8 | 10 |
| RGB Image [10] | 23.4% | 76.6% | 25.2% | 26.3 | 7.8 |
| one sequence(200 frames) | | | | | |
| RGB-D Video | **12.7**% | **87.3**% | **18.9**% | 18.1 | 14.1 |
| Depth two-stage | 14.3% | 85.7% | 19.8% | 18.8 | 13 |
| RGB Video [13] | 15.8% | 84.2% | 22.0% | 19.8 | 11.1 |
| RGB Video [14] | 17.5% | 82.5% | 23.1% | 19.6 | 13 |
| RGB Image [10] | 24.7% | 75.3% | 28.2% | 25.8 | 10.8 |

compared with the dense segmentation Figure 2 and Figure 4. A structured placement did not perform as well, indicating that the random structure of the pattern gives a good spatial coverage of the neighborhood. Also incorporating depth and color in spatio-temporal graph makes our method much more efficient compared to 'Depth two-stage' method inspired by [16], since we only apply spectral clustering once to obtain the segmentation whereas the other method needs to perform spectral clustering twice. The effect of the Nyström method is demonstrated in Figure 5 and Figure 3. We see that the approximation degrades gracefully with the sparsity, and that qualitatively sensible segmentations are achieved at a sampling rate of 50% or lower. Also the computation time for spectral clustering increases drastically with respect to the number of pixels. However, we can observe that choosing a smaller size window for processing the frames in conjunction with choosing a larger sample size gives better accuracy and less computation time.

## VI. CONCLUSION

In this work, we have proposed a novel framework for spatio-temporal segmentation from RGBD videos that incorporates a rich set of appearance, motion, object-part and depth cues while maintaining feasible computation on a single core.

The system employs appearance cues based on low-level information such as intensity and color, while also incorporating high-level information from a deformable parts model (DPM). In this way, we are able to compute *object centric* spatio-temporal segmentations based on semantic object detection. One interesting area of future work based on the proposed idea is to use this technique to segment and track the object detected by DPM and improve the performance of object detection where DPM fails to detect the object due to different factors in the scene.

Several computational innovations are incorporated in order to make feasible a dense spatio-temporal segmentation on a single core. First, a Nyström approximation is used based that samples pixels from the dense optic-flow graph. This strategy combines the computational efficiency of the Nyström approximation with the accuracy of an intelligent sampling strategy that accounts for the temporal structure of the problem. Additionally, we have incorporated a structured connectivity pattern for construction of the graph Laplacian. This strategy balances the high accuracy achieved by a connectivity with high spatial extent with the computational imperative of a similarity matrix with low degree. The resulting method has a computational efficiency sufficiently low to enable computation on a single core. Future work will be aimed at decreasing the computation time and use an online approach for streaming the video data.

The proposed framework has been validated using two benchmark datasets: the NYU Depth Dataset [22] and the TUM RGBD Dataset [27]. On both datasets, results show across the board improvements on a wide range of performance metrics. These include error and accuracy percentages, the degree of over-segmentation, and the number of objects extracted. In summary, we have shown that the addition of depth information, a structured connectivity pattern for the graph Laplacian, and a deformable parts model all conspire to substantially improve spatio-temporal segmentation of RGB-D video.

## REFERENCES


[1] A. Abramov, K. Pauwels, J. Papon, F. Worgotter, and B. Dellen. Depth-supported real-time video segmentation with the kinect. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 457–464. IEEE, 2012.

[2] G. Alenyà, S. Foix, and C. Torras. Using tof and rgbd cameras for 3d robot perception and manipulation in human environments. *Intelligent Service Robotics*, 7(4):211–220, 2014.

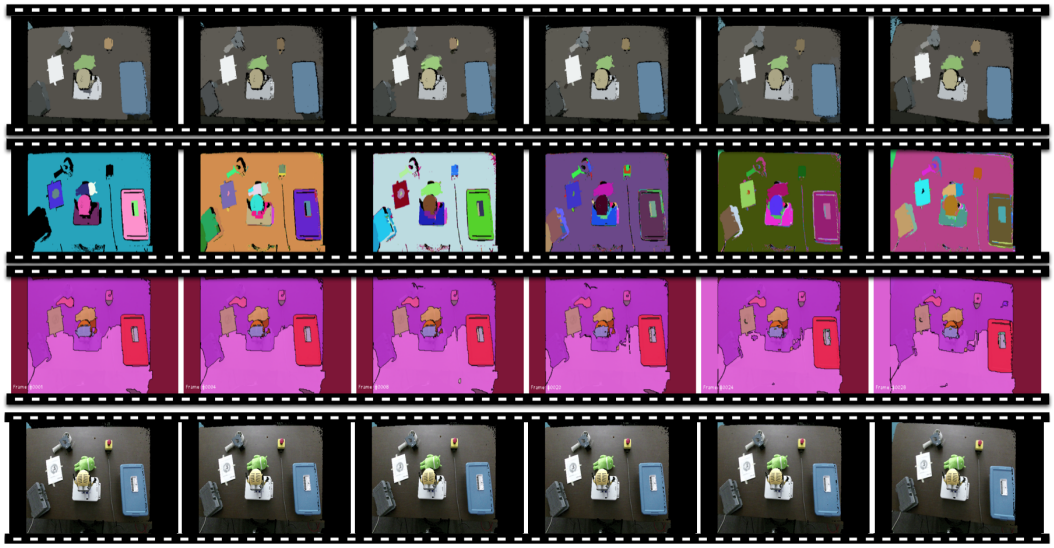

Fig. 8. Segmentation results by the proposed method for a sequence of size 6 shown in the last row. The first row: our temporal RGB-D, the second row: only RGB [10] and the third row: temporal RGB [14]. The proposed method has the best performance in terms of precision and recall of object pixels (scored using motion-verified object boundaries).


[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *ECCV*, pages 282–295. 2010.

[5] J. Chang, D. Wei, and J. W. Fisher III. A video representation using temporal superpixels. In *CVPR*, 2013.

[6] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1124–1131. IEEE, 2005.

[7] Daniel DeMenthon and Remi Megret. Spatio-temporal segmentation of video by hierarchical mean shift analysis. Technical Report UMIACS-TR-2002-68, University of Maryland, College Park, 2002.

[8] C. Erdogan, M. Paluri, and F. Dellaert. Planar segmentation of RGBD images using fast linear fitting and Markov chain Monte Carlo. In *Computer and Robot Vision (CRV)*, pages 32–39, 2012.

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[10] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, Sept. 2004.

[11] A. Fossati, J. Gall, H. Grabner, X. Ren, and K. Konolige. *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*. Springer, 2012.

[12] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214–225, Feb 2004.

[13] M. Ghafarianzadeh, M. B. Blaschko, and G. Sibley. Unsupervised spatio-temporal segmentation with sparse spectral clustering. In *BMVC*, 2014.

[14] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. *IEEE CVPR*, 2010.

[15] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Computer Vision–ECCV 2014*, pages 345–360. Springer, 2014.

[16] S. Hickson, S. Birchfield, I. Essa, and H. Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 344–351. IEEE, 2014.

[17] D. Holz and S. Behnke. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Proceedings of the 12th International Conference on Intelligent Autonomous Systems (IAS)*, Jeju Island, Korea, June 2012.

[18] H. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 32(8):951–970, 2013.

[19] M. Li, X.-C. Lian, J. T. Kwok, and B.-L. Lu. Time and space efficient spectral clustering via column sampling. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2297–2304. IEEE, 2011.

[20] C. Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, MIT, 2009.

[21] C. McManus, W. Churchill, W. Maddern, A. Stewart, and P. Newman. Shady dealings: Robust, long- term visual localisation using illumination invariance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[22] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[23] S. R. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[24] A. Richtsfeld, T. Mörwald, J. Prankl, J. Balzer, M. Zillich, and M. Vincze. Towards scene understanding - object segmentation using rgbd-images. In *Proc. of the 17th Computer Vision Winter Workshop (CVWW 2012)*, 2012.

[25] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, Aug. 2000.

[26] J. Strom, A. Richardson, and E. Olson. Graph-based segmentation for colored 3D laser point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[28] N. Sundaram and K. Keutzer. Long term video segmentation through pixel level spectral clustering on GPUs. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Nov 2011.

[29] R. Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[30] D. Weikersdorfer, A. Schick, and D. Cremers. Depth-adaptive supervoxels for rgb-d video segmentation.

[31] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.